CERES Software Bulletin 10-01

April 30, 2010

AMI Deliveries Policies

Tammy Ayers (Tammy.O.Ayers@nasa.gov)
Denise Cooper (Denise.L.Cooper@nasa.gov)
Scott Zentz (Scott.M.Zentz@nasa.gov)

1.0 Introduction

In order to ensure software deliveries from the CERES DMT to the *AMI* system at the Langley ASDC will work in a production environment with a dual architecture system, a few guidelines need to be instituted. Also, to take advantage of this multiprocessor environment provided by *AMI* all subsystem teams should endeavor to migrate their software to both the x86 and P6 platforms. At this time the number of x86 resources in the SCF and Operations cluster are much greater than P6 resources. Also, the current thinking is that the next generation cluster environment may/will contain all x86 machines. Further at issue is enabling the ASDC SIT and Production teams to take full advantage of SGE and cluster resources, so this will be part of the guidelines as well.

2.0 Mandatory Procedures for *AMI* Deliveries

In addition to the normal procedures for making deliveries to CERES CM, subsystem personnel will also need to follow the instructions in this bulletin for deliveries to *AMI*.

2.1 $HOSTTYPE Environment Variable

The $HOSTTYPE environment variable will need to be set for other functions (see below) to work properly. $HOSTTYPE should be set by one of the following two ways.

1. setenv HOSTTYPE `arch` (these backticks are necessary for the command to work)

2. chomp(my $arch_type = `arch`); (these backticks are necessary for the command to work)
   $ENV{'HOSTTYPE'} = $arch_type;

HOSTTYPE is equal to “x86_64” on the x86 platform and “ppc64” on the P6 platform.

2.2 Binaries and the bin Directory

The name of the binary executable needs to be standardized to some degree so that an outside observer can tell what the file is from which PGE it came and for what architecture it is intended. The best way to tell that a file is a binary executable is to simply make the executable name end with “.exe”. How users signify the PGE name is really up to them, but the recommendation is that the PGE name be included in the name of the executable file. The requirement is again that

an outside observer will be able to tell which PGE the file belongs to. The final requirement is that within the executable name the $HOSTTYPE environment variable be used to differentiate between the architectures. This way the bin directory structure does not need to be altered to accommodate two architectures.

Examples: CER1.1P8_x86_64.exe
CER4.1-4.1P3.retrieval_ppc64.exe

2.3 Scripts and the rcf Directory

There should be no reason that multiple scripts need to be used because of the two architectures. The $HOSTTYPE variable or the `arch` command can be used for any branching via conditionals when the script needs to perform differently per the architecture. The values of $HOSTTYPE and the `arch` command are "ppc64" for P6 and "x86_64" for x86. Since all architecture differentiation can be done inside of scripts, there is no need to split the rcf directory.

NOTE: The pair of backquotes or backticks (`) around the arch command is necessary for the command to work.

2.4 Source Code, Makefiles, and the src/lib Directory

If, in order to accommodate the two different architectures, two different versions of the PGE or library source code are deemed necessary by the subsystem team, implementation of this solution must be approved by the CERES DM Lead.

If approved, then the recommended first approach is to make two versions of the source code file(s) in question and differentiate them with the $HOSTTYPE variable. Two Makefiles may also need to be created with the file names containing $HOSTTYPE such that both will compile all the non-altered code plus their architecture specific source files. One Makefile may also be used as long as the source file names are differentiated by $HOSTTYPE.

Examples: Makefile_$HOSTTYPE
moa_io_$HOSTYPE.f90

If, however, the code changes are deemed significant enough, a second approach is to split the src and/or lib directories for the PGE. Then, you can use any name you wish for the new directory paths, as long as it is clear from an outside observer's point of view with which architecture the code is associated. Within the name of the ".a" file the $HOSTTYPE environment variable must be used to differentiate between the architectures.

Examples:

mkdir `arch`
cd $CERESHOME/sarb/CER5.4/src/`arch`, where `arch` is either ppc64 or x86_64

lib_sarb_$HOSTTYPE.a
moa_lib_$HOSTTYPE.a

2.5 Data Directory

The data directory structure should mirror that of the ASDC DPO, which is found at /ASDC_archive/CERES, whether or not the data files are stored in the DPO. For example,

DPO directory:

/ASDC_archive/CERES/SSF/Terra_FM1_Edition2/2009/04

Cloud's data directory:

$CERESHOME/CERES/clouds/data/SSF/Terra_FM1_Edition2/2009/04

2.6 PCF Generation

PCFin files will no longer be created with the start of *AMI* deliveries. The PCF generation logic will now need to include finding all necessary input files and verifying their existence. In lieu of PCFin files a new file will be created during PCF generation; it will be essentially a log file. This log file will contain allthe environment variables used, all the mandatory files not found, all the optional files not found, and all the files found and used. Templates of this new code have been made and tested and will be made available as needed.

In order to allow the SIT and Production teams to use SGE and the cluster resources, delivered scripts cannot be allowed to prompt the user for information. Interactive scripts cannot be used effectively in SGE and a cluster environment. The PROD environment variable is set in production to "Yes", so a script can key off of this to turn off interactivity in scripts.

PCF scripts need to either succeed or fail. Either the script finds all the mandatory files it needs or it doesn't. If it succeeds, the PCF script needs to exit with a 0. If it fails to find all the mandatory files it needs, then the PCF script must fail with 203. All the information one needs concerning the files that will be used or the files that could not be found for a given PGE run will be listed in the companion log file.

2.7 Expected Output

Generally speaking you should not need to differentiate by architecture for expected output, since output from a PGE MUST create scientifically equivalent output in order for it to be considered usable. So an output file from P6 should be equivalent to output from X86. In order to test this, the test suites software must compare variables using a delta rather than a direct comparison of real numbers or straight binary diffs. Appropriate "delta" values are determined or approved by the Science Team. For PGEs that use proper test suites that compare with deltas, no changes are required. For PGEs lacking proper test suites it is likely the files will not compare exactly because of differences with architecture or compiler. The options available in

this case are to re-write the test suites to use deltas or to split the directory path of the expected output.

3.0 $PROD Environment Variable

The $PROD environment variable can be set to "Yes" or "No"; "No" for development, CM testing, and SIT testing, and "Yes" for production. $PROD can be used for anything where you want to do something differently in development and testing from the way it's done in production. For instance, $PROD can be used to determine where to look for input data. If your test input data is not available in the DPO during development and testing keying off of $PROD allows you to point to the data directory in your subsystem directory structure instead.

4.0 Acronym List

ASDC	Atmospheric Science Data Center
CERES	Clouds and the Earth's Radiant Energy System
DM	Data Management
DMT	Data Management Team
DPO	Data Products Online
PCF	Process Control File
PGE	Product Generation Executive
SGE	Sun Grid Engine
SIT	Science Integration and Test